matCUDA installation guide

64-bits WINDOWS OS

version 1.0

Index

1. Introduction

matCUDA is a free open-source c++ library that aims to facilitate the development of C++ codes that involves generalized vectors and matrix operations with high performance. An instance of the herein presented Array class represents an array (vector or matrix), whose needed memory is automatically allocated at the class instantiation. Mathematical operators are overloaded, providing intuitive ways of perform algebraic operations. Besides, there are some algebraic functions implemented. Most of the operations and functions are CUDA-implemented, in order to take advantage of the high performance provided by the use of GPUs onto computation.

The Array class is the most important class contained in matCUDA library and is responsible to interface between the user and the other classes, responsible for automatic allocation/deallocation of memory, data transfer to and from the GPU, GPU and CPU operations, etc. It has some mathematical operators overloaded, in a way that a multiplication of two instances of Array is performed through the operator "*", and its sum by "+", and most of its functionalities are implemented as public members.

2. Requirements

To use or develop, you will need.

- At least one 64-bits CUDA-capable GPU (with CUDA capacity ≥ 2.0). Check if you have a CUDA-capable GPU here or here.
- CUDA toolkit ≥ 7.0. Can be downloaded from here.
- BOOST library ≥ 1.53. Can be downloaded from here.
- A C/C++ IDE (integrated development environment). This work was formerly developed upon Microsoft Visual Studio Ultimate 2012, but others may apply.

3. I want to use

For those who just want to use it without develop. On the matCUDA github repository, download file "libs+test.zip".

- The file you should include in your project is "inc/matCUDA.h"
- The library you should link is "x64/Release/matCUDA.lib"
- Properly link the CUDA and BOOST parts to your project. On how to do it, refer to section 6 (**How to configure a Visual Studio Project to use matCUDA**).

4. I want to test (STRONGLY RECOMMENDED)

For those who want to test before use it. On the matCUDA github repository, download file "libs+test.zip".

- The file you should execute is "x64/Release/matCUDA tests.exe". It is supposed to run the unit tests on your machine.
- If it does not happen, go to section 5 (**I want to use and develop**) ("I want to use and develop").

5. I want to use and develop

For those who just want to use and develop it as well. Download the repository and unzip it.

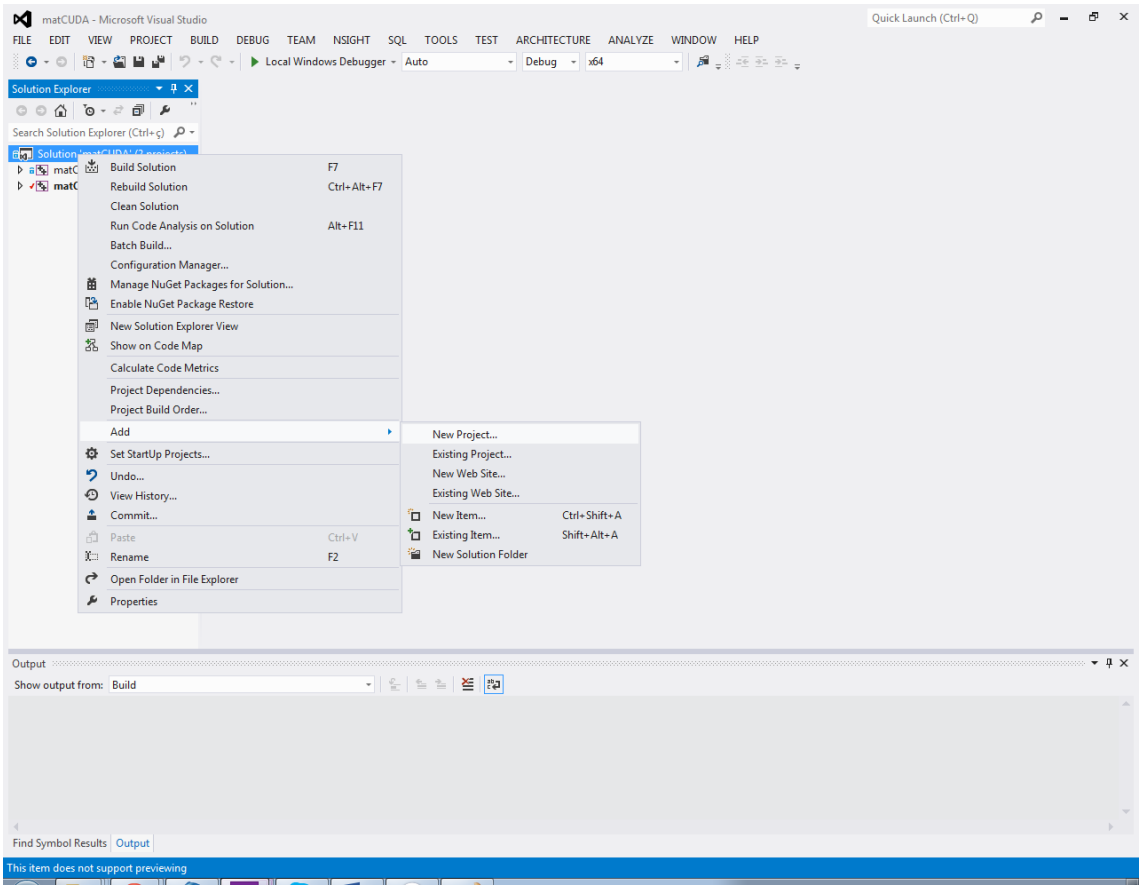
- If you have Visual Studio 2012 or later, the projects are already setup*, and you just open "VS2012 solution/matCUDA.sln". The development project ("matCUDA") and the test project ("matCUDA tests") will be loaded. Just Build it.
- Otherwise, you will have to properly set up your projects. On how to do it, refer to section 6 (**How to configure a Visual Studio Project to use matCUDA**)

*This is considered in case you installed version 1.53 of BOOST library under "C:\local\boost_1_53_0".

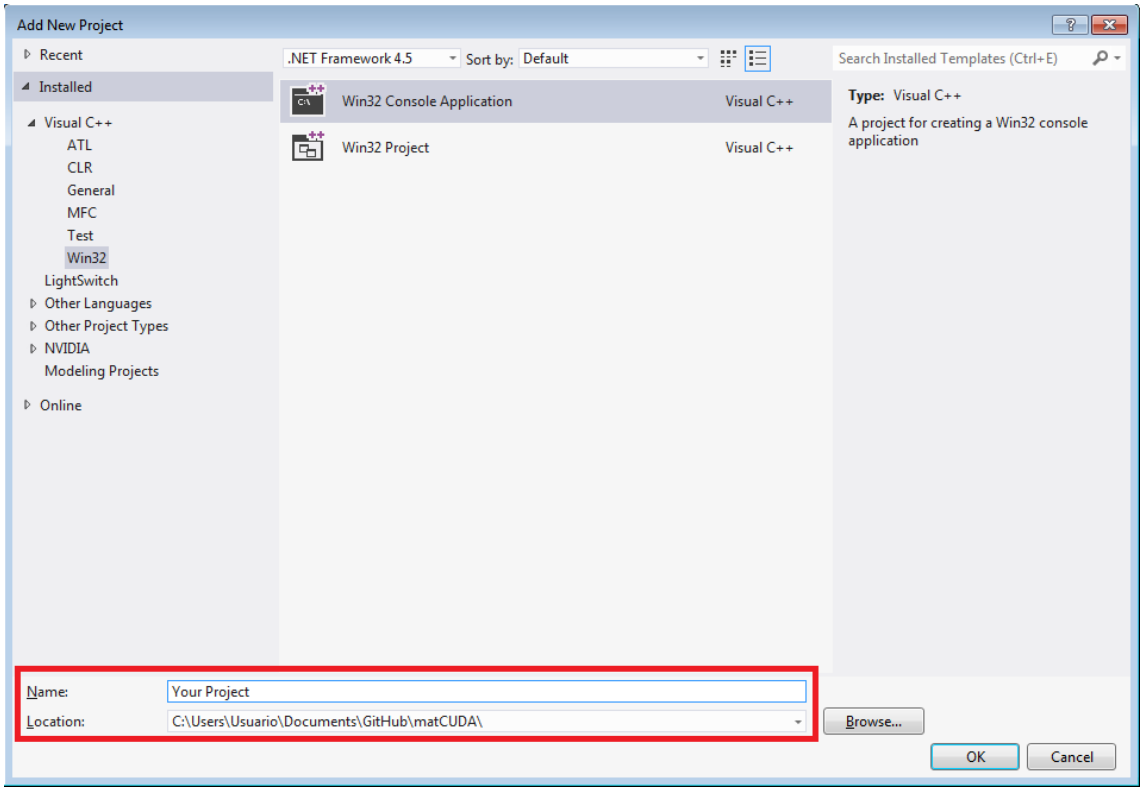
6. How to configure a Visual Studio Project to use matCUDA

The user is advised to add his/her new project on the Solution provided (under "VS2012 solution\matCUDA.sln"). It is no requirement, although the possibility to set up dependencies among the projects under the same Solution favor the development and usage of the matCUDA library on the same Solution, as will be shown. It is assumed that you have BOOST version 1.53 under "C:\local\boost_1_53_0".

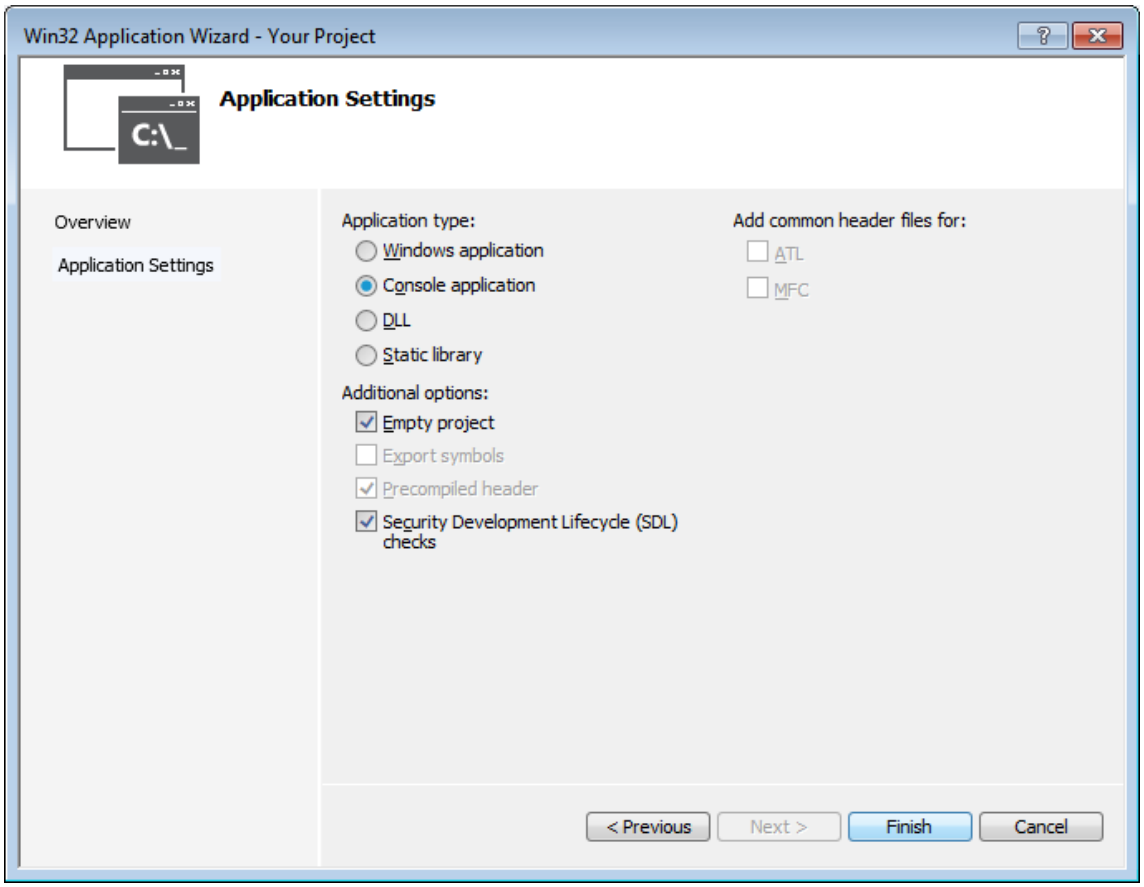
6.1. Right-click the solution name, then "Add -> New Project"



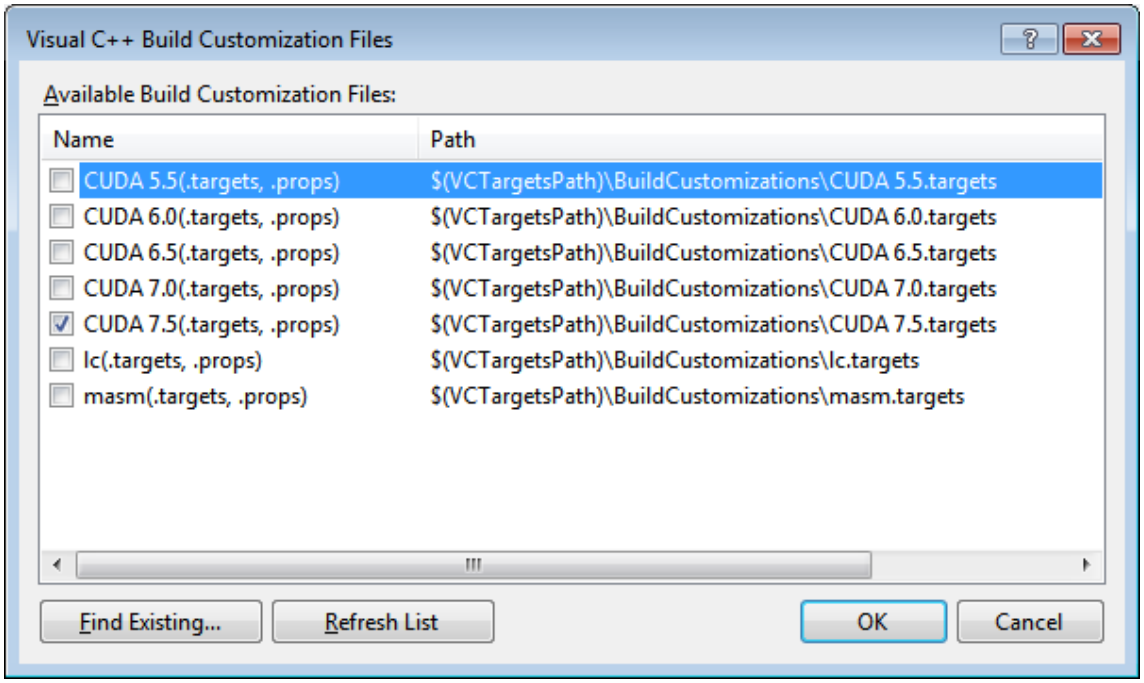
6.2. On the "Add New Project" window, choose "Win32 Console Application", and choose the Name and Location of this new project. *Hint: if you choose the Default Location, it will create under "VS2012 solution". Create under "matCUDA", so all the 3 projects will be on the same level of folders.*



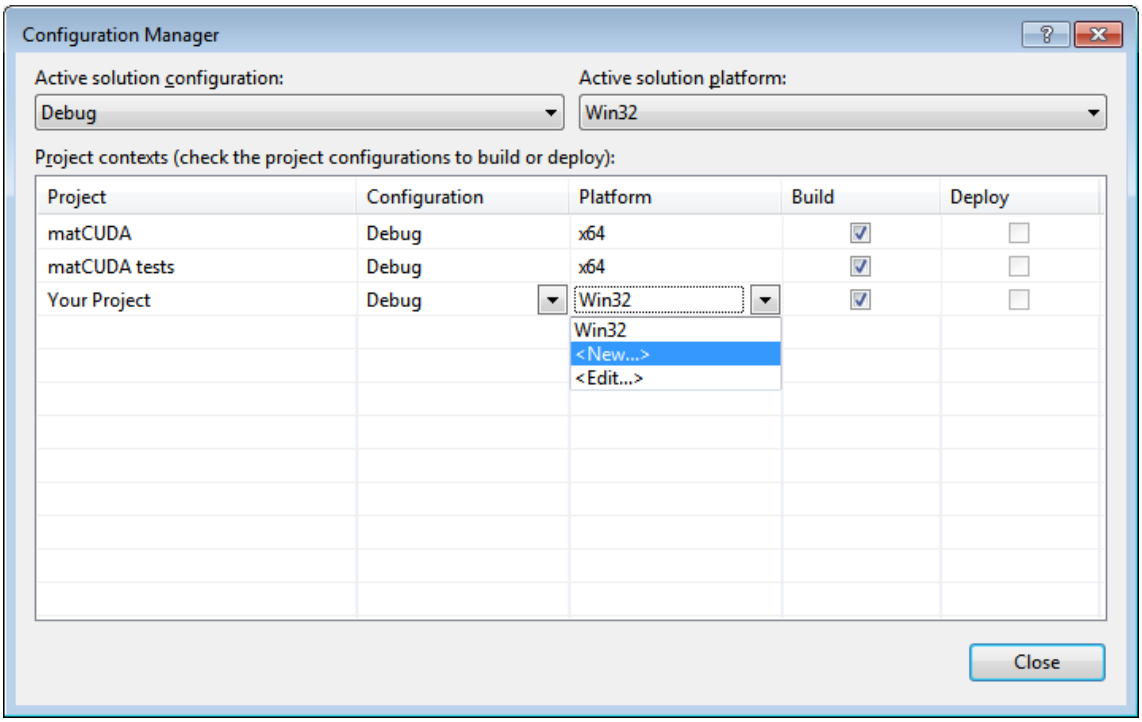
6.3. Click “OK”, “Next” and check the box “Empty Project”. Click “Finish”.



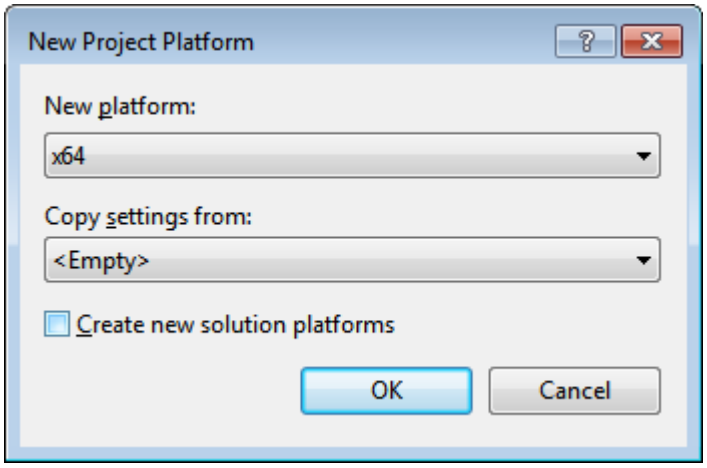
6.4. Right-click your new project name, then “Build Customizations…” and check the CUDA 7.0 (or higher) box. Click “OK”.



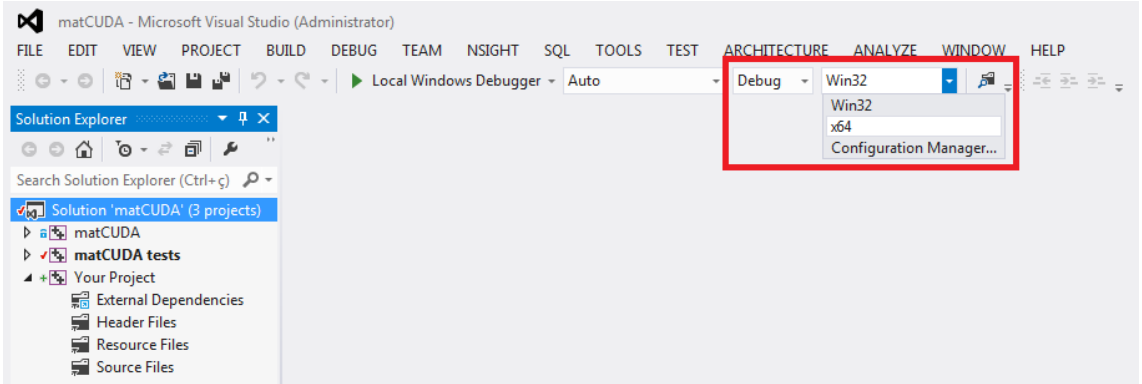
6.5. Right-click the Solution, then “Configuration Manager”, and select “<New…>” under “Platform” of Your Project. Also, check the “Build” box.



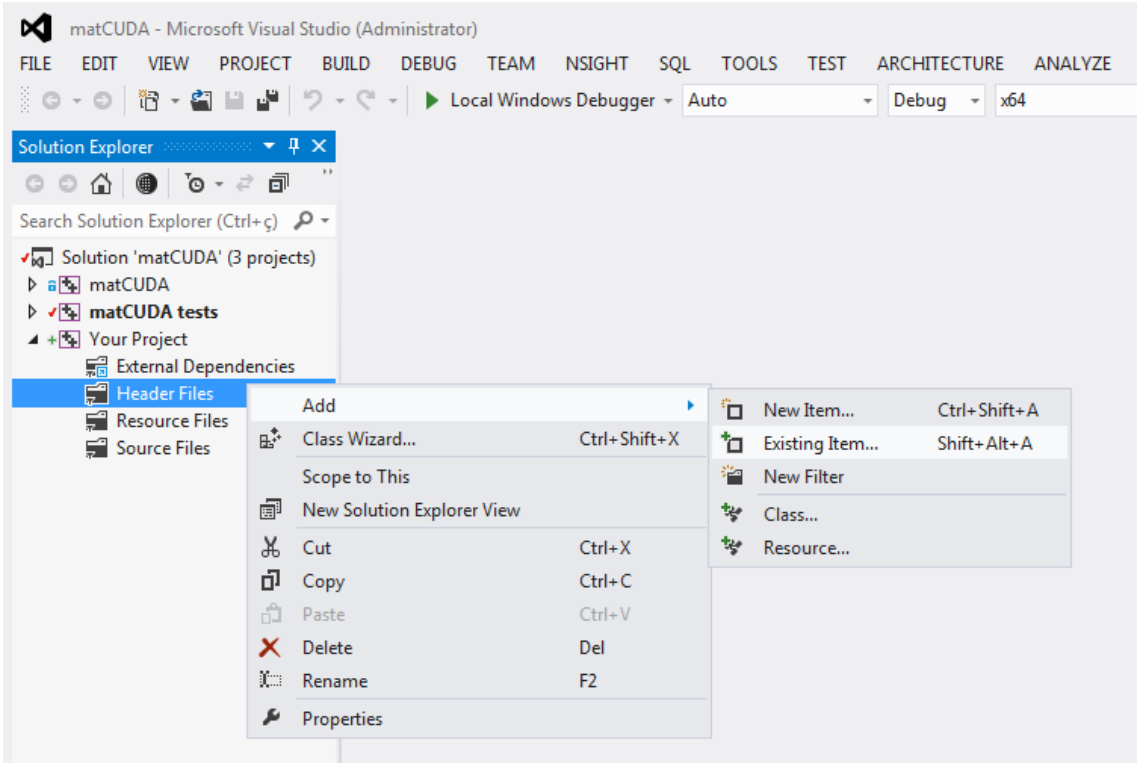
6.6. Choose “x64” under “New Platform” and “<Empty>” under “Copy settings from”. Click “OK”, then “Close”.



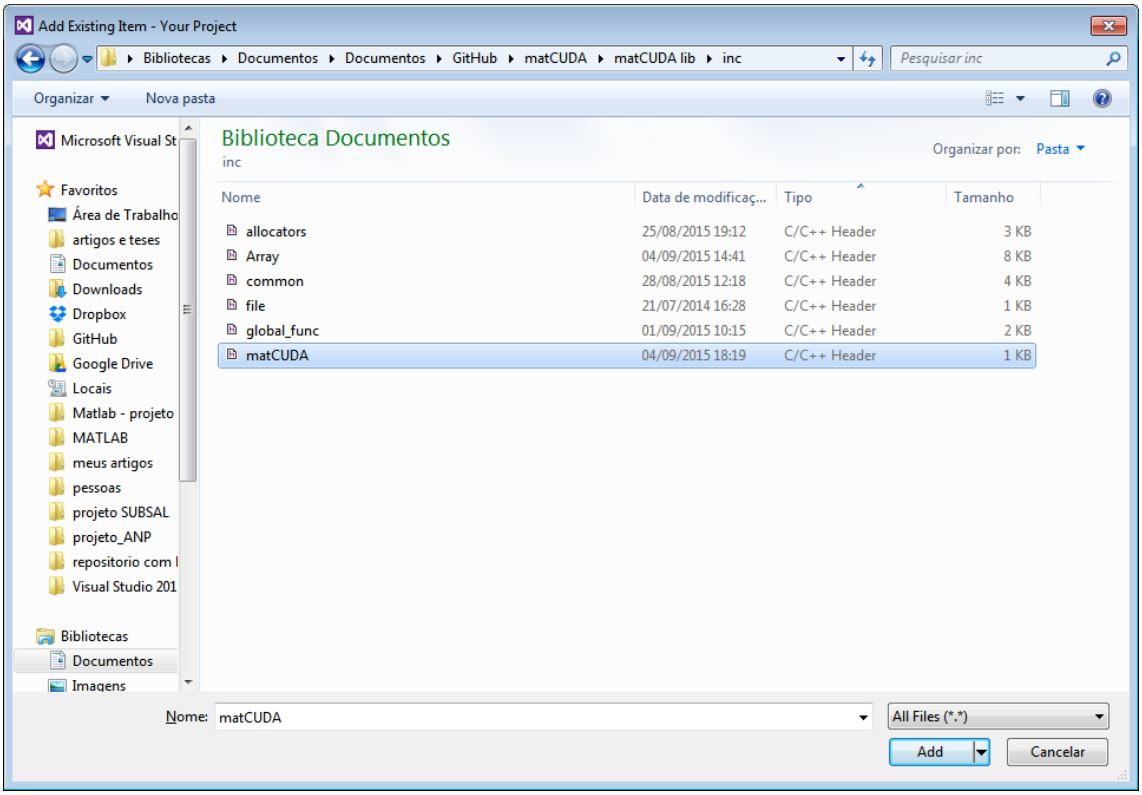
6.7. Change your “Solution Platform” from “win32” to “x64”.



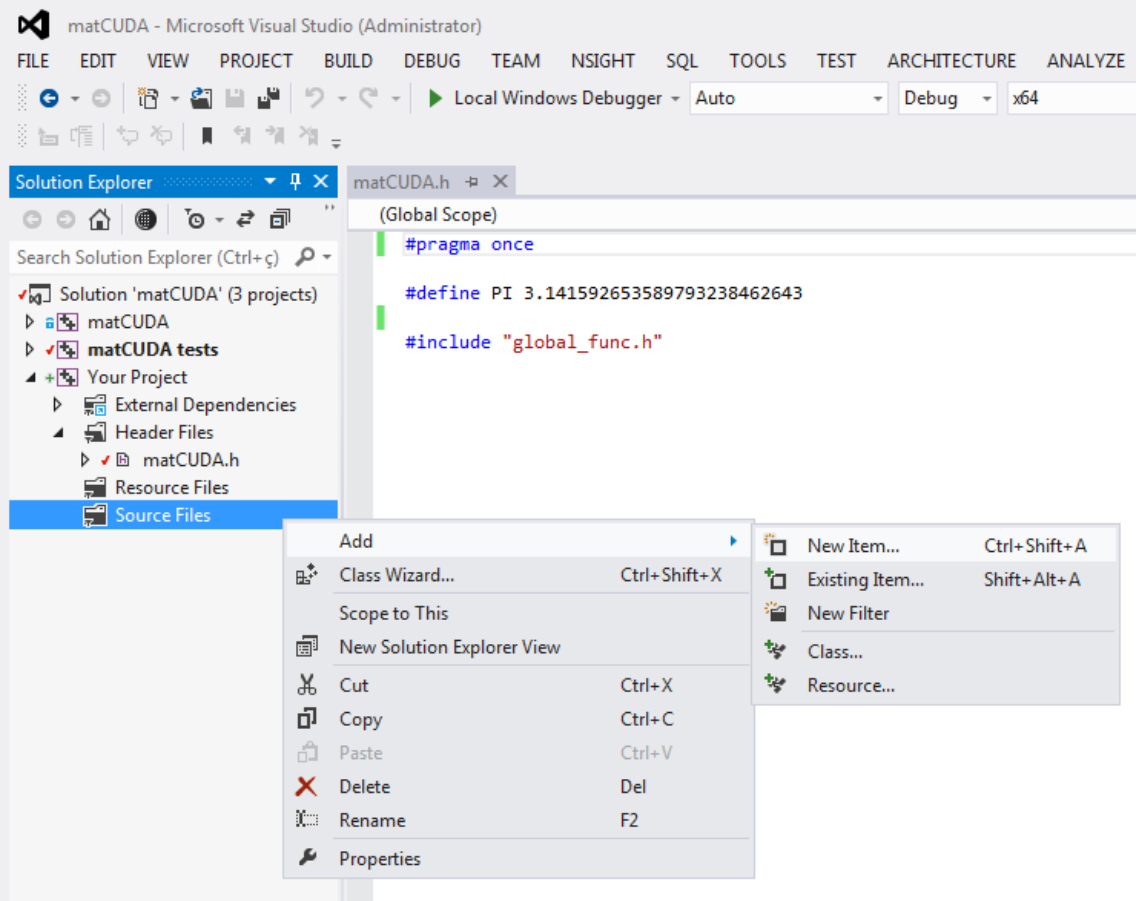
6.8. Right click “Header Files” under “Your Project”, then “Add -> Existing Item”.



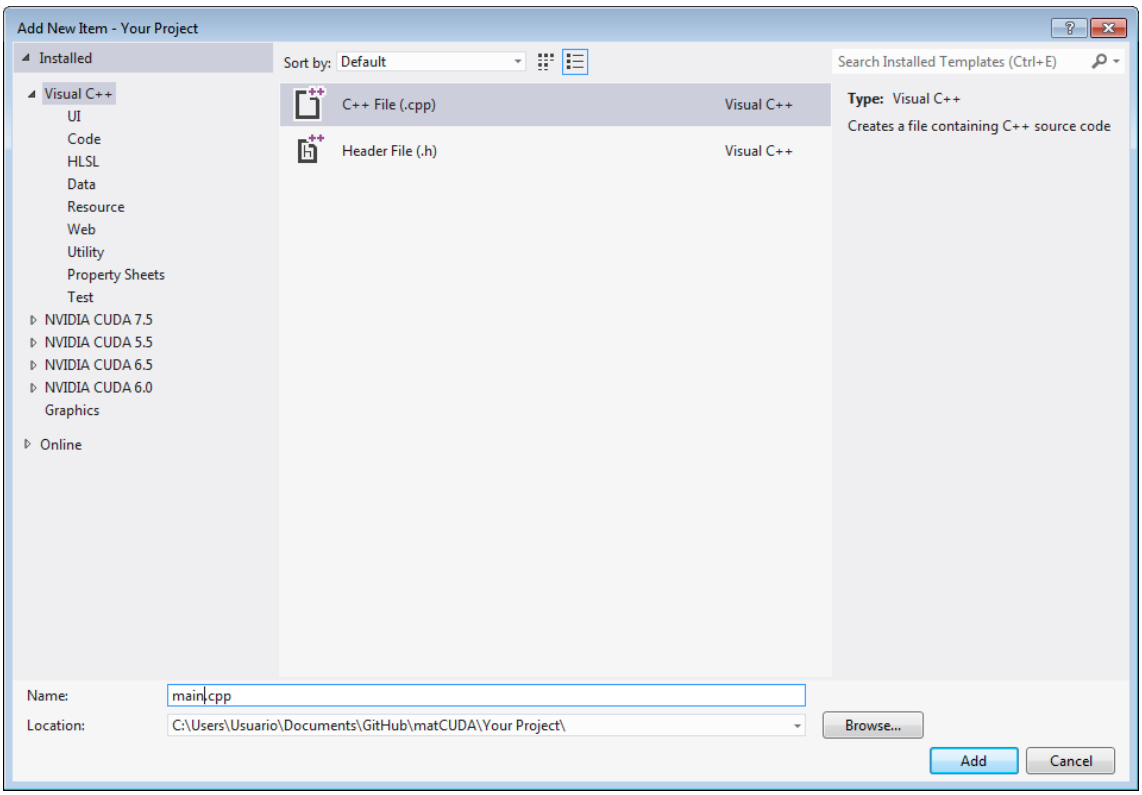
6.9. Choose “matCUDA lib\inc\matCUDA.h”, then “Add”



6.10. Right click “Source Files” under “Your Project”, then “Add -> New Item”.



6.11. Name it whatever you want, as “main.cpp”, for example. Then “Add”.



6.12. Now right-click “Your Project”, then “Properties”, and change the following on the “Configuration Properties”. Make sure that the selected “Platform” is “x64”.

6.12.1. Under “VC++ Directories -> Include Directories” add “../matCUDA lib/inc;C:\local\boost_1_53_0;”. *(You have to set the path/to/matCUDA.h and the path/to/BOOST)*

6.12.2. Under “VC++ Directories -> Library Directories” add “C:\local\boost_1_53_0\lib64-msvc-11.0;”

6.12.3. Under “Linker -> Input -> Additional Dependencies”, add the following libraries: “..\VS2012 solution\x64\Debug\matCUDA.lib;cusparse.lib;cusolver.lib;cufft.lib;curand.lib;cublas.lib;cudart.lib;”

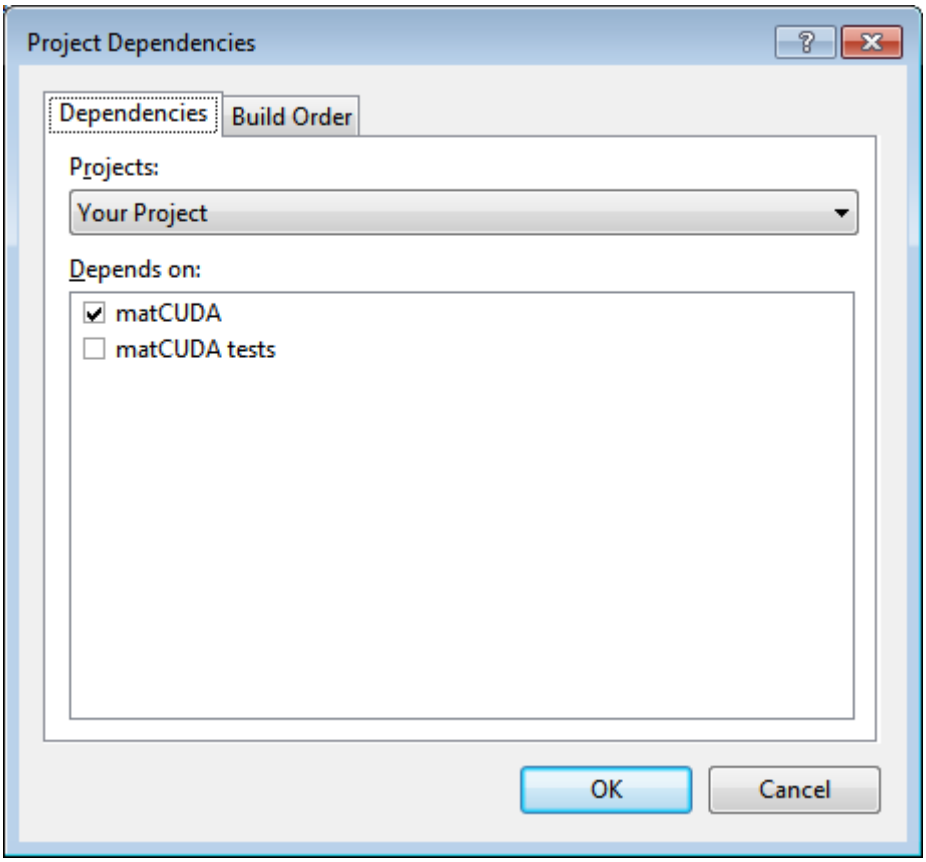
6.12.4. Under "C/C++ -> Code Generation -> Runtime Library", select "Multi-threaded Debug DLL (/MDd)

6.12.5. Under "C/C++ -> Command Line", add "-D_ITERATOR_DEBUG_LEVEL=0"

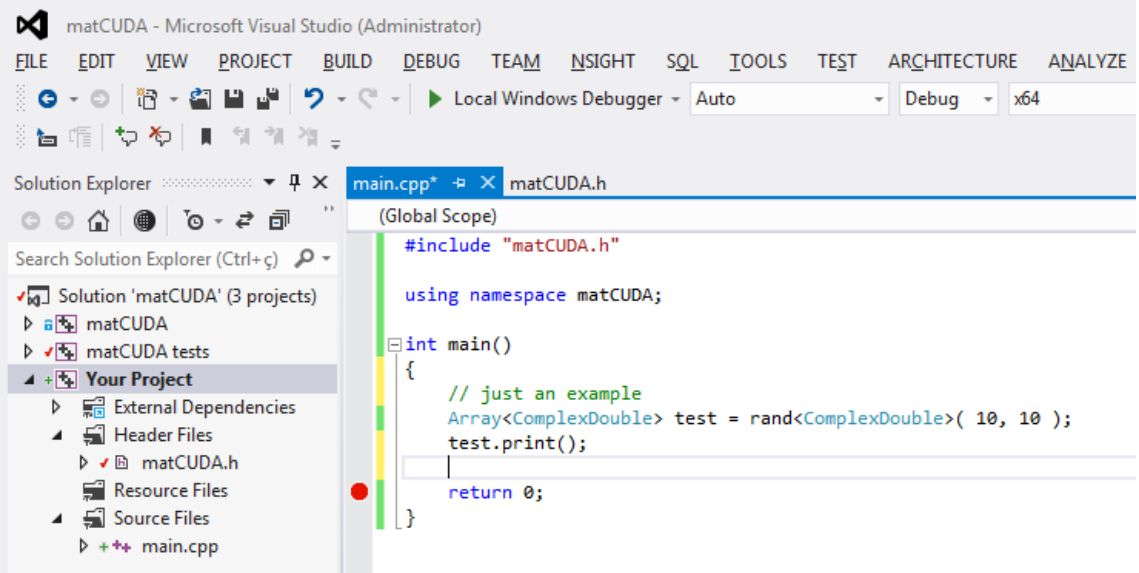
6.12.6. Under "Linker -> Debugging -> Generate Debug Info", select "Yes (/DEBUG)"

6.13. Right-click “Your project -> Set as StartUp Project”

6.14. Right-click “Your project -> Project Dependencies”, and check the “matCUDA” box, so it will be build every time you build Your Project.



6.15. On your “main.cpp” file, add as the image below.



6.16. On the Menu, click on “DEBUG -> Start Debugging”. If it Builds and Run, you will see the black DOS screen fill up with random numbers.

If you want to work with the “Release” version of the Compiler, instead of “Debug”, repeat the procedures above to the “Release” version.

7. Example

Below is a piece of code that provides an overall idea of how matCUDA works, since the creation of an instance of the Array class that represents a vector or matrix and some operations.

```
#include "matCUDA.h"

using namespace matCUDA;

int main()
{
	size_t size = 1024;

	// creates ComplexDouble-type Array object with size random elements - 
vector
	Array<ComplexDouble> v1 = rand<ComplexDouble>( size );

	// creates ComplexDouble-type Array object with size x size random 
elements - matrix
	Array<ComplexDouble> m1 = rand<ComplexDouble>( size, size );

	// multiplies m1 times v1 and stores in v2
	Array<ComplexDouble> v2 = m1*v1;

	// creates m2 with dimensions of m2
	Array<ComplexDouble> m2( m1.getDim(0), m1.getDim(1) );

	// check if determinant of m1 is different from 0. if so, inverts m1 and 
stores in m2
	if( m1.determinant() != ComplexDouble(0,0) )
		m2 = m1.invert();
```

```
        // check is m1 times m2 equals identity matrix
        bool equalIdentity = m1*m2 == eye<ComplexDouble>( size );

        return 0;
}
```
